Descrizione UML Iniziale
Gruppo GC6 (Rigamonti-Rogora-Sarbu)

1. Introduzione

Abbiamo scelto di utilizzare il Model-View-Controller prestando particolare attenzione nel separare la parte di logica (situata nel Controller) dallo stato di una partita (situata nel Model). Nell'UML iniziale è Descritta la parte del Model, i cui metodi sono chiamati dal Controller attraverso un'istanza di GameModel.

2. Model

2.1 GameModel

Viene utilizzato come interfaccia per il controller al fine di chiamare metodi più complessi. In particolare, i metodi semplici che possono essere chiamati direttamente da una classe istanziata in GameModel non vengono implementati nuovamente, come ad esempio per ottenere la posizione di Madre Natura il Controller chiama GameModelInstance.getMotherNature().getPosition(), che restituisce un intero. Se la modalità passata al costruttore è Expert, i componenti della Expert Mode vengono inizializzati.

2.2 Player & SchoolBoard

Player contiene al suo interno un'istanza di SchoolBoard e un'istanza del mazzo di Assistenti, di cui è possibile scegliere il Mago (WIZARD_1, WIZARD_2, …). Tutti i componenti sono muniti di Getter e Setter relativi. SchoolBoard presenta invece una lista di studenti per l'entrata, 5 stack per la Dining Room (uno per ogni colore), una lista di professori, una lista di torri e una CoinSupply che viene inizializzata solo se la modalità di gioco è Expert. Anch'essi sono tutti muniti di getter e setter.

2.4 IslandGroup & IslandTile

Le IslandGroup sono inizializzate con una sola IslandTile al loro interno. Nel momento in cui viene effettuata una Join(), che riceve come parametro un altro IslandGroup, le torri vengono controllate. Nel caso in cui il colore corrisponda, viene eliminato il secondo IslandGroup e le IslandTiles al suo interno vengono annesse al primo. Il resto dei metodi consiste in getter e setter, anche per le NoEntryTiles, che possono essere aggiunte in Expert Mode.

2.4 Character

La classe Character presenta lo stato attuale del personaggio. Essa è costituita dal Nome (descritto dall'enum CharacterType), dal costo attuale e da un Boolean, effectIsUsed, che permetterà al controller di vedere se l'effetto di un personaggio passivo è stato attivato (come quello del Centauro). Se il nome corrisponde a uno dei 4 personaggi che necessitano di pedine aggiuntive, esse vengono istanziate. L'effetto viene gestito nel Controller, poiché dipende dalla fase del turno in cui viene attivato.

3. Note Aggiuntive

Nel diagramma sono stati omessi alcuni getter e setter per rendere il modello più comprensibile. Il Controller non viene rappresentato nell'UML del modello. Esso contiene più istanze di GameModel, identificate da un "codice partita" al fine di implementare la funzionalità di istanze multiple di gioco. Il controller viene descritto da una FSM il cui stato è salvato in un'apposita classe, chiamata GameLobby, che è implementata separatamente. Ad ogni GameLobby è associato uno e un solo GameModel.